1. What is GraphQL and how does it differ from REST?

GraphQL is a query language for APIs that allows clients to request specific data, providing a more efficient and flexible approach to data fetching compared to REST. Unlike REST, which uses multiple endpoints, GraphQL offers a single endpoint for all data requests, reducing over-fetching and under-fetching by letting clients specify exactly what they need.

2. Explain the concept of a GraphQL schema.

"A GraphQL schema is a blueprint that describes the structure of the API, including types, queries, mutations, and subscriptions. It acts as a contract between the client and server, ensuring consistent data exchange.

3. What are queries in GraphQL? Provide an example of a simple query.

"Queries in GraphQL are the mechanism for fetching data from the server, allowing clients to specify exactly what data they need. For example, a simple query to fetch a user's name and email would look like ***{ user { name email } }."***

4. What are mutations in GraphQL? Write a mutation to create a new user.

"Mutations in GraphQL are the mechanism for modifying data on the server, allowing clients to perform operations like creating, updating, or deleting data. For example, a simple mutation to create a new user would look like ***mutation { createUser(name: "John Doe", email: "john@example.com") { id name email } }."***

5. Describe the purpose of resolvers in GraphQL.

"Resolvers in GraphQL are functions that handle fetching and manipulating data in response to client queries and mutations. They map schema fields to data sources, ensuring the correct data is returned and can implement custom logic like authentication and authorization."

6. How do you handle errors in GraphQL?

"Errors in GraphQL can be handled using the errors field in the response, which provides detailed information about any issues that occurred. Additionally, custom error messages can be used to give more context to the client, and logging errors is crucial for debugging and monitoring purposes."

7. What is the role of the GraphQL server?

"The GraphQL server acts as the intermediary between client requests and data sources, processing queries and mutations by executing the necessary resolvers. It ensures data consistency and handles authentication and authorization, providing a seamless data-fetching experience."

8. Explain the difference between queries and subscriptions in GraphQL.

"Queries in GraphQL are static requests for data retrieval, allowing clients to fetch specific data at a single point in time. Subscriptions, on the other hand, are used for real-time data updates, requiring a persistent connection, typically using WebSockets."

9. Write a GraphQL query to fetch a list of books with their titles and authors.

"To fetch a list of books with their titles and authors, you can use a GraphQL query ***like { books { title author } }.*** *This query ensures that you retrieve only the necessary fields, optimizing performance and reducing data transfer."*

10. How can you implement pagination in a GraphQL query?

"To implement pagination in a GraphQL query, you can use the first and after arguments to fetch a specific number of items and navigate through the dataset. This approach helps manage large datasets efficiently by retrieving data in smaller, manageable chunks."

11. What are fragments in GraphQL? Provide an example of how to use them.

"Fragments in GraphQL are reusable units of query logic that help reduce redundancy by grouping fields that can be reused in multiple queries. For example, you can create a fragment to fetch a user's name and email with fragment userFields on User { name email }."

12. Explain the concept of input types in GraphQL and provide an example.

"Input types in GraphQL are special objects used to pass structured data into queries and mutations, helping to validate and organize input data. For example, an input type for a new user could be defined as input ***NewUserInput { name: String! email: String! }****."*

13. Write a GraphQL mutation to update a user's email address.

"To update a user's email address, you can use a GraphQL mutation like ***mutation { updateUser(id: "1", email: "newemail@example.com") { id email } }.*** *This mutation specifies the user's ID and the new email address, ensuring the correct user is updated with the provided information."*

14. How do you implement authentication in a GraphQL API?

"To implement authentication in a GraphQL API, you can use middleware to verify user tokens, such as JWT. Libraries like Passport.js can be integrated to handle authentication, ensuring that resolvers check the authentication status before processing requests."

15. What is the purpose of the `@deprecated` directive in GraphQL?

"The @deprecated directive in GraphQL is used to mark fields or methods that should no longer be used, signaling to clients that these parts of the API are outdated. This helps ensure a smoother transition when updating or refactoring the API by clearly communicating which elements may be removed in the future."

16. Write a GraphQL query that uses variables to fetch a user by ID.

"To fetch a user by ID using variables in GraphQL, you can define a query with a variable and use it in the query. For example, the query ***query getUser($id: ID!) { user(id: $id) { name email } }*** *allows you to dynamically fetch a user's name and email by their ID."*

17. Explain the concept of batching and caching in GraphQL.

"Batching in GraphQL combines multiple requests into a single request to reduce server load, while caching stores frequently requested data to improve response times and reduce redundant data fetching. Both techniques enhance performance and efficiency in GraphQL APIs."

18. How do you handle nested queries in GraphQL? Provide an example.

"Nested queries in GraphQL allow you to request related data in a hierarchical structure, improving efficiency by fetching all necessary data in a single request. For example, a nested query to fetch a user with their posts and comments would look ***like { user { name posts { title comments { text } } } }."***

19. Write a GraphQL subscription to listen for new messages in a chat application.

"To listen for new messages in a chat application, you can use a GraphQL subscription like subscription { newMessage { id content sender } }. This subscription sets up a real-time connection to receive updates whenever a new message is sent."

20. What are the benefits of using GraphQL over traditional REST APIs?

"GraphQL allows clients to request specific data, reducing over-fetching and under-fetching, which optimizes performance. Additionally, it provides a single endpoint for all data requests, simplifying API management and supporting real-time updates through subscriptions."

21. Describe how to implement a custom scalar type in GraphQL.

"To implement a custom scalar type in GraphQL, you define its name and validation logic, extending GraphQL's built-in types to handle specialized data formats. For example, a custom scalar for a date type might include serialization and parsing functions to ensure proper date formatting."

22. Write a GraphQL query that fetches a user's posts along with the comments for each post.

"To fetch a user's posts along with the comments for each post, you can use a nested GraphQL query. For example, the ***query { user { posts { title comments { text } } } }*** *retrieves the posts and their associated comments efficiently."*

23. How do you optimize performance in a GraphQL API?

"To optimize performance in a GraphQL API, you can implement query optimization techniques to reduce server load and improve response times. Additionally, using caching mechanisms to store frequently requested data and implementing batching to combine multiple requests into a single request can significantly enhance performance."

24. Explain the concept of schema stitching in GraphQL.

"Schema stitching in GraphQL is the process of combining multiple GraphQL schemas into a single unified schema, allowing seamless integration of data from various microservices. This approach helps create a cohesive API from disparate data sources, enhancing the flexibility and scalability of your application."

25. Write a GraphQL mutation to delete a post by its ID.

"To delete a post by its ID, you can use a GraphQL mutation like mutation ***{ deletePost(id: "1") { id } }.*** *This mutation specifies the post ID to ensure the correct post is deleted and returns the ID of the deleted post."*

26. What do you know by a non-null type modifier in GraphQL?

In GraphQL, a non-null type modifier is used to specify an argument as non-null value. If you pass null for a non-null argument, the GraphQL server will send a validation error. The exclamation mark "!" is used to mark an argument as non-null.

27. What do you mean by aliases in GraphQL?

In GraphQL, the alias is used to change the field name of the query. It facilitates you to rename the result of a field according to yourself.

For example:

```
{
classA : employee(id : String){
name
}
classB : employee(id: String){
name
}
}
```

In the above query, we have used an alias to use a different name for two 'employees.' Without alias, you will get an error in the result.

28. What is Authentication and Authorization in GraphQL?

Authentication and AuthorizationAuthorization are the processes used in services. Sometimes people get confused in these two terms and exchange their definitions for each other.

Authentication: Authentication is a process that is used to claim an identity. Authentication is done when you want to log in to a service with a username and password. Here, you have to authenticate yourself. In GraphQL, Authentication can be implemented with common patterns such as OAuth. OAuth is an open protocol that is used to allow secure AuthorizationAuthorization in a simple and standard method from web, mobile, and desktop applications.

Authorization: On the other hand, AuthorizationAuthorization is a process that is used to give permission rules that specify the access rights of individual users and user groups to certain parts of the system. For authorization implementation in GraphQL, it is recommended to delegate any data access logic to the business logic layer and not handle it directly.

29. How to do Error Handling in GraphQL?

It is easy to see the error in GraphQL. A successful GraphQL query is supposed to return a JSON object with a root field called "data." If your request query fails or partially fails, you will see a second root field called "errors" in the response. See the below example

Example:

```
{
  "data": { ... },
```

3. "errors": [...]
4. }

Note: Sometimes, the request query fails or partially fails if the user requesting the data doesn't have the right access permission.

30. What do you know about GraphiQL?

GraphiQL is used to provide UI representation for GraphQL. It is an in-browser IDE used to explore GraphQL and make GraphQl usage easy. GraphQL supports real-time error highlighting, so you can see and handle errors easily.

31. Spring Boot GraphQL example?

Spring Boot - GraphQL Integration - GeeksforGeeks

gopi582/spring-boot-graphql-mongodb